

A New Approach for Improving Pseudorandomness of Pseudorandom Sequences with Applications

Lequan Min

Department of Information and Computer Science, School of Mathematics and Physics,
University of Science and Technology Beijing, Beijing 100083, China
13501029489@163.com

Abstract

Based on the Golomb's pseudorandomness assumptions on idea pseudorandom sequences and FIPS 140-2 pseudorandomness test, this paper first presents a new approach for improving the pseudorandomness of pseudorandom sequences. Second, using a generalized synchronization theorem, and three chaotic maps constructs one 8-dimensional chaotic generalized synchronization system (8DCGSS). Then using the 8DCGSS designs a chaotic pseudorandom number generator (CPRNG). The keyspace of the CPRNG is larger than 2^{1117} . Third, using FIPS 140-2 pseudorandomness test criterions and generalized FIPS 140-2 pseudorandomness test criterions measures, respectively, the pseudorandomness of the keystreams with length 20 000, 100 000 and 1 000 000 generated via the CPRNG, an Matlab PRNG, an RC4 algorithm, and an m-sequence with period $2^{20} - 1$, and the corresponding improved keystreams by our approach. The results show that the presented approach can increase significantly the pseudorandomness of the keystreams generated by the four PRNGs. The key streams generated by the m-sequence do not have sound pseudorandomness when the lengths of the key streams are less than 100 000.

keywords: Improving pseudorandomness, Golomb's pseudorandomness assumptions, Pseudorandom sequences, Chaotic pseudorandom number generator, RC4 algorithm, m-Sequence, FIPS 140-2 test

Contents

1	Introduction	1
2	An approach for improving pseudorandomness of pseudorandom sequences	2
2.1	A new chaotic generalized synchronization system	13
2.2	A chaotic pseudorandom number generator	14
2.3	Improvements and measures of pseudorandomness for keystreams generated from 4 PRNGs . .	17
2.4	Improvements and measures of pseudorandomness for keystreams with length $1e5$ generated from 4 PRNGs	23
2.5	Improvements and measures of pseudorandomness for keystreams with length 10^6 generated from 4 PRNGs	24
3	Conclusions	27

1 Introduction

Pseudorandom number sequences are useful in many applications such as simulations of physical systems (Example: Binder and Heermann[1], and Senate[2]), entertainment (Example: Wegenkittl[3]), computer simulation

(Example: Knuth[4]), stochastic optimization methods (such as simulated annealing), watermarking for image authentication and particularly cryptography (Example: Ferguson[5]).

The randomness of idea pseudorandom number sequences should satisfy Golomb's assumptions Golomb[6]. That a Poor PRNG generates low quality pseudorandom number sequences will be harmful, especially leak the prevented information in secrete information transformations.

Statistical tests on PRNGs play a fundamental role in measuring certain properties of pseudorandom number sequences. FIPS 140-2 pseudorandomness test criterions introduced by the National Institute of Standards and Technology (NIST) of the USA ([7]) is a well-known public-domain statistical testing packages for PRNGs.

Hight quality pseudorandom number sequences are very useful in different fields. In aim to obtain better pseudorandom sequence sources, this paper presents a new approach for improving the randomness of pseudorandom sequences base on the Golomb's idea pseudorandom sequence assumptions [6] and the FIPS 140-2 pseudorandomness test criterions. Using a generalized synchronization theorem Zang *et al.*[8] and the Henon map, the logistic map and a tube map (Yang *et al.*[10]) constructs a new 8-dimensional chaotic generalized synchronization system (8DCGSS). Then using the 8DCGSS designs a CPRNG. Third, using FIPS 140-2 pseudorandomness test criterion and generalized FIPS 140-2 (GFIPS 140-2) pseudorandomness test criterions Min *et al.*[9] measures the pseudorandomness of the keystreams with length 20 000 generated by the CPRNG, an Matlab PRNG, an RC4 algorithm, and an m-sequence with period $2^{20} - 1$, and measures the improved keystreams via our approach. Then using the two randomness test criterions measures the pseudorandomness of the keystreams with length 100 000 and length 10 000 000 generated by the four PRNGs, and the corresponding improved keystreams by our approach, respectively. The results show that the presented approach can increase significantly the pseudorandomness of the keystreams generated by the four PRNGs. The key streams generated the m-sequence with period $2^{20} - 1$ do not have sound pseudorandomness in the meanings of the Golomb's pseudorandomness assumptions when the lengths of the key streams are less than 100 000.

The rest of the paper is organized as follows. Section 2 proposes the approach for improving pseudorandomness of pseudorandom sequences. Section 3 is divided to three subsections. Subsection 3.1 presents a new 8DCGSS. Subsection 3.2 designs a CPRNG based on the 8DCGSS. Subsection 3.3 introduces the GFIPS 140-2, and uses FIPS 140-2 test and GFIPS 140-2 test measures the pseudorandomness of the keystreams with 20 000 bit lengths generated via the CPRNG, the Matlab PRNG, the RC4 algorithm, and the m-sequence, and the corresponding improved keystreams via our approach. Subsections 3.4 and 3.5 use the methods in Subsection 3.3 to study the pseudorandomness of the key streams with 100 000 bit and 1 000 000 bit lengths generated by the four PRNGs, respectively. Finally, some concluding remarks are presented in Section 4.

2 An approach for improving pseudorandomness of pseudorandom sequences

Based on the Golomb's assumptions on the randomness that pseudorandom sequences should satisfy [6] and FIPS 140-2 randomness test criteria, this section proposes an approach for improving the randomness of pseudorandom sequences.

Now using a sequence S with length l of 20 000 bits as an example describes our approach. This approach consists of eight steps:

(1) **Improving the balance property of S .** According to the Golomb's assumption, an idea pseudorandom sequence with even length l should have $l/2$ zeros and $l/2$ ones. Denote the numbers of the 0's and the 1's in S by l_0 and l_1 , respectively.

(a) If $d = l_0 - l_1 > 0$. Denote $m = \text{fix}(l_0/\text{fix}(d/2))$, where fix is a Matlab command that $\text{fix}(X)$ rounds the elements of X to the nearest integers towards zero. Then substitute continuously 0's to 1's by m interval in the sequence S until all absent 1's have been supplied.

(b) If $d = l_1 - l_0 > 0$. Denote $m = \text{fix}(l_1/\text{fix}(d/2))$. Then substitute continuously 1's to 0's by m interval in the sequence S until all absent 0's have been supplied.

(2) Improving the uniformity of 1-runs.

Denote l_0 and l_1 to be the numbers of 0's 1-runs and 1's 1-runs, respectively.

(a) If $d = l_0 - 2500 \geq 0$. Find the segments which have the following forms:

$$J01 = [1011011], J02 = [1101101]. \quad (1)$$

Denote d_0 to be the number of forms (1) and $m = \text{fix}(2d_0/d)$. Then replace continuously segments (1) to the forms:

$$D01 = [1001111], D02 = [1111001] \quad (2)$$

by interval m until all excess 0's have been deleted.

(b) If $d = 2500 - l_0 \geq 0$. Find the segments which have the following forms:

$$J01 = [1001111], J02 = [1111001]. \quad (3)$$

Denote d_0 to be the number of forms (3) and $m = \text{fix}(2d_0/d)$. Then replace continuously segments (3) to the forms:

$$D01 = [1011011], D02 = [1101101] \quad (4)$$

by interval m until all absent 0's have been supplied.

(c) If $d = l_1 - 2500 \geq 0$. Find the segments which have the following forms:

$$J11 = [0100100], J12 = [0010010] \quad (5)$$

Denote d_1 to be the number of forms (5) and $m = \text{fix}(2d_1/d)$. Then replace continuously segments (5) to the forms:

$$D11 = [0110000], D12 = [0000110] \quad (6)$$

by interval m until all excess 1's have been supplied.. (d) If $d = 2500 - l_1 \geq 0$. Find the segments which have the following forms:

$$J11 = [0110000], J12 = [0000110]. \quad (7)$$

Denote d_1 to be the number of forms (7) and $m = \text{fix}(2d_1/d)$. Then replace continuously segments (7) to the forms:

$$D11 = [0100100], D12 = [0010010] \quad (8)$$

by interval m until all absent 1's have been supplied..

Remark: Practically, we can always find enough Jij 's defined by (1), (3), (5) and (7).

(3) Improving the uniformity of 2-runs.

Denote l_0 and l_1 to be the numbers of 0's 2-runs and 1's 2-runs, respectively.

(a) If $d = l_0 - 1250 \geq 1$. Find the segments which have the following forms:

$$J01 = [100100001], J02 = [100001001]. \quad (9)$$

Denote d_0 to be the number of forms (9) and $m = \text{fix}(d_0/d)$. Then replace continuously segments (9) to the forms:

$$D01 = [100010001], D02 = [100010001] \quad (10)$$

by interval m until all excess 0's 2-runs have been delete if possible. If there do not exit enough J01's and J02's defined by (9) to reduce the number of zero 2-runs, we need to find the segments which have the following forms:

$$J03 = [11100111001], J04 = [10011100111] \quad (11)$$

and then change continuously them to the forms:

$$D03 = [11111100001], D04 = [10000111111] \quad (12)$$

until all surplus zero 2-runs have been deleted if possible.

(b) If $d = 1250 - l_0 \geq 1$. Find the segments which have the following form:

$$J01 == [100010001]. \quad (13)$$

Denote d_0 to be the number of forms (13) and $m = \text{fix}(d_0/d)$. Then replace continuously segments (13) to the form:

$$D01 = [100100001] \quad (14)$$

by interval m until all absent zero 2-runs have been supplied if possible. If we cannot find enough J01's defined by (13) to reduce the number of zero 2-runs, we need to find the continuously segments which have the following forms:

$$J02 = [11111100001], J03 = [10000111111] \quad (15)$$

and then change continuously them to the forms:

$$D02 = [11100111001], D03 = [10011100111] \quad (16)$$

until all absent zero 2-runs have been generated if possible.

(c) If $d = l_1 - 1250 \geq 1$. Find the segments which have the following form:

$$J11 = [011011110], J12 = [011110110]. \quad (17)$$

Denote d_1 to be the number of forms (17) and $m = \text{fix}(d_1/d)$. Then replace continuously segments (17) to the form:

$$D11 = [011101110], D12 = [011101110] \quad (18)$$

by interval m until all excess one 2-runs have been deleted if possible. . If we cannot find enough J01's and J02's defined by (17) to reduce the number of one 2-runs, we need to find the segments which have the following forms:

$$J13 = [00011000110], J14 = [01100011000] \quad (19)$$

and then change continuously them to the forms:

$$D11 = [000000111110], D12 = [00001111000] \quad (20)$$

until all surplus one 2-runs have been deleted if possible.

(d) If $d = 1250 - l_1 \geq 1$. Find the segments which have the following form:

$$J11 = [0111011110], \quad (21)$$

Denote d_1 to be the number of form (21) and $m = \text{fix}(d_1/d)$. Then replace continuously segments (21) to the form:

$$D11 = [011011110] \quad (22)$$

by interval m until all absent one 2-runs have been generated if possible. If we cannot find enough J01's defined by (21) to increase the number of one 2-runs, we need to find the segments which have the following forms:

$$J12 = [00000011110]; J13 = [01111000000], \quad (23)$$

and then change continuously them into the forms:

$$D12 = [00011000110], D13 = [01100011000] \quad (24)$$

until all absent one 2-runs have been generated if possible.

(4) **Improving the uniformity of 3-runs.**

Denote l_0 and l_1 to be the numbers of zero 3-runs and one 3-runs, respectively.

(a) If $d = l_0 - 625 \geq 1$. Find the segments which have the following forms:

$$J01 = [10001000001], J02 = [10000010001]. \quad (25)$$

Denote d_0 to be the number of form (25) and $m = \text{fix}(d_0/d)$. Then replace continuously segments (25) to the form:

$$D01 = [10000100001], D02 = [10000100001] \quad (26)$$

by interval m until all excess zero 3-runs have been generated if possible. If we cannot find enough J01's and J02's defined by (25) to reduce the number of zero 3-runs, we need to find the segments which have the following forms:

$$J03 = [111100011110001], J04 = [100011110001111], \quad (27)$$

and then change continuously them into the forms:

$$D03 = [111111110000001], D04 = [100000011111111] \quad (28)$$

until all surplus zero 3-runs have been deleted if possible.

(b) If $d = 625 - l_0 \geq 1$. Find the segments which have the following form:

$$J01 = [10000100001]. \quad (29)$$

Denote d_0 to be the number of form (29) and $m = \text{fix}(d_0/d)$. Then replace continuously segments (29) to the form:

$$D01 = [10001000001] \quad (30)$$

by interval m until all absent zero 3-runs have been generated if possible. If we cannot find enough J01's defined by (29) to increase the number of zero 3-runs, we need to find the segments which have the following forms:

$$J02 = [111111110000001], J03 = [100000011111111] \quad (31)$$

and then change continuously them to the forms:

$$D02 = [100011110001111]; D03 = [100011100011111], \quad (32)$$

until all absent zero 3-runs have been generated if possible.

(c) If $d = l_1 - 625 \geq 1$. Find the segments which have the following form:

$$J01 = [01110111110], J02 = [01111101110] \quad (33)$$

Denote d_1 to be the number of form (33) and $m = \text{fix}(d_1/d)$. Then replace continuously segments (33) to the form:

$$D01 = [01111011110], D02 = [01111011110] \quad (34)$$

by interval m until all surplus one 3-runs have been deleted if possible. If we cannot find enough J01's and J02's defined by (33) to reduce the number of one 3-runs, we need to find the segments which have the following forms:

$$J04 = [111111110000001], J05 = [100000011111111] \quad (35)$$

and then change continuously them to the forms:

$$D03 = [100011110001111]; D04 = [100011100011111], \quad (36)$$

until all surplus one 3-runs have been deleted if possible.

(d) If $d = 625 - l_1 \geq 1$. Find the segments which have the following form:

$$J01 == [01111011110]. \quad (37)$$

Denote d_1 to be the number of form (37) and $m = \text{fix}(d_1/d)$. Then replace continuously segments (37) to the form:

$$D01 = [01111011110] \quad (38)$$

by interval m until all absent one 3-runs have been generated if possible. If there do not exit enough J01's defined by (37) to increase the number of one 3-runs, we need to find the segments which have the following form:

$$J02 = [11111111000000], J03 = [0000001111111], \quad (39)$$

and then change continuously them to the forms:

$$D02 = [11110001111000], D03 = [00011110001111], \quad (40)$$

until all absent one 3-runs have been generated if possible.

(5) Improving the uniformity of 4-runs.

Denote l_0 and l_1 to be the numbers of zero 4-runs and one 4-runs, respectively.

(a) If $d = l_0 - 313 \geq 1$. Find the segments which have the following form:

$$J01 = [1000010000001], J02 = [1000000100001], \quad (41)$$

Denote d_0 to be the number of form (41) and $m = \text{fix}(d_0/d)$. Then replace continuously segments (41) to the form:

$$D01 = [1000001000001]; D02 = [1000001000001], \quad (42)$$

by interval m until all surplus zero 4-runs have been deleted if possible. If we cannot find enough J01's and J02's defined by (41) to reduce the number of zero 4-runs, we need to find continuously segments which have the following forms:

$$J03 = [1111100001111100001]; J04 = [1000011111000011111] \quad (43)$$

and then change them to the forms:

$$D03 = [1111111111000000001], D04 = [1000000001111111111] \quad (44)$$

until all surplus zero 4-runs have been deleted if possible.

(b) If $d = 313 - l_0 \geq 1$. Find the segments which have the following form:

$$J01 = [1000001000001]. \quad (45)$$

Denote d_0 to be the number of form (45) and $m = \text{fix}(d_0/d)$. Then replace segments (45) to the form:

$$D01 = [1000010000001] \quad (46)$$

by m interval. If we cannot find enough J01's defined by (45) to increase the number of zero 4-runs. In this case, we need to find continuously segments which have the following forms:

$$J02 = [1111111111000000001], J03 = [1000000001111111111] \quad (47)$$

and then change them to the form:

$$D02 = [1111100001111100001], D03 = [1000011111000011111] \quad (48)$$

until all absent zero 4-runs have been supplied if possible.

(c) If $d = l_1 - 313 \geq 1$. Find the segments which have the following forms:

$$J01 = [0111101111110], J02 = [011111011110]. \quad (49)$$

Denote d_1 to be the number of form (49) and $m = \text{fix}(d_1/d)$. Then replace continuously segments (49) to the forms:

$$D01 = [0111110111110], D02 = [0111110111110] \quad (50)$$

by interval m until all surplus one 4-runs have been deleted if possible. If we cannot find enough J01's and J02's defined by (49) to reduce the number of one 4-runs, we need to find continuously segments which have the following forms:

$$J03 = [0000011110000011110], J04 = [0111100000111100000] \quad (51)$$

and then change them to the forms:

$$D03 = [0000000000111111110], D04 = [0111111110000000000] \quad (52)$$

until all surplus one 4-runs have been deleted if possible.

(d) If $d = 313 - l_1 \geq 1$. Find the segments which have the following form:

$$J11 == [0111110111110]. \quad (53)$$

Denote d_1 to be the number of form (53) and $m = \text{fix}(d_1/d)$. Then replace continuously segments (53) to the form:

$$D11 = [0111101111110]. \quad (54)$$

If we cannot find enough J11's defined by (53) to increase the number of one 4-runs, we need to find the segments which have the following forms:

$$J12 = [0000000000111111110], J13 = [0111111110000000000] \quad (55)$$

and then change continuously them to the form:

$$D12 = [0000011110000011110], D13 = [0111100000111100000] \quad (56)$$

until all absent one 4-runs have been generated if possible.

(6) Improving the uniformity of 5-runs.

Denote l_0 and l_1 to be the numbers of zero 5-runs and one 5-runs, respectively.

(a) If $d = l_0 - 156 \geq 1$. Find the segments which have the following forms:

$$J01 = [10000010000000], J02 = [00000001000001]. \quad (57)$$

Denote d_0 to be the number of form (57) and $m = \text{fix}(d_0/d)$. Then replace continuously segments (57) to the forms:

$$D01 = [10000001000000], D02 = [00000010000001] \quad (58)$$

by interval m until all excess zero 5-runs have been deleted if possible. If we cannot find enough J01's and J02's defined by (57) to reduce the number of zero 5-runs, we need to find continuously the segments which have the following forms:

$$J03 = [100000110000000], J04 = [000000011000001] \quad (59)$$

Denote d_{00} to be the number of form (59) and $m_1 = \text{fix}(d_{00}/(d - d_0))$. Then replace continuously segments (59) to the forms:

$$D03 = [00000011000000], D04 = [000000110000001] \quad (60)$$

by interval m_1 until all surplus zero 5-runs have been deleted if possible. If we cannot find enough J03's and J04's defined by (59) to reduce the number of zero 5-runs, we need to find continuously the segments which have the following forms:

$$J05 = [100000110000000], J06 = [000000011000001]. \quad (61)$$

Denote d_{000} to be the number of form (61) and $m_2 = \text{fix}(d_{000}/(d - d_0 - d_{00}))$. Then replace continuously segments (61) to the forms:

$$D05 = [100000011000000], D06 = [000000110000001] \quad (62)$$

by interval m_2 until all surplus zero 5-runs have been deleted if possible. If we cannot find enough J05's and J06's defined by (61) to reduce the number of zero 5-runs, we need to find continuously the segments which have the following form:

$$J07 = [1000001000001], \quad (63)$$

Denote d_{0000} to be the number of form (63) and $m_3 = \text{fix}(2d_{0000}/(d - d_0 - d_{00} - d_{000}))$. Then replace continuously segments (63) to the form:

$$D07 = [11000001000001] \quad (64)$$

by interval m_3 until all surplus zero 5-runs have been deleted if possible.

(b) If $d = 156 - l_0 \geq 1$. Find the segments which have the following forms:

$$J01 = [10000001000000], J02 = [00000010000001]. \quad (65)$$

Denote d_0 to be the number of form (65) and $m = \text{fix}(d_0/d)$. Then replace continuously segments (65) to the forms:

$$D01 = [10000010000000], D02 = [00000001000001] \quad (66)$$

by interval m until all absent zero 5-runs have been supplied if possible. If we cannot find enough J01's and J02's defined by (65) to increase the number of zero 5-runs, we need to find continuously segments which have the following forms:

$$J03 = [100000011000000], J04 = [000000110000001]. \quad (67)$$

Denote d_{00} to be the number of form (67) and $m_1 = \text{fix}(d_{00}/(d - d_0))$. Then replace continuously segments (67) to the forms::

$$D03 = [100000110000000], D04 = [000000011000001] \quad (68)$$

by interval m_1 until all absent zero 5-runs have been generated if possible. If we cannot find enough J03's and J04's defined by (67) to increase the number of zero 5-runs, we need to find continuously segments which have the following forms:

$$J05 = [1000000111000000], J06 = [0000001110000001]. \quad (69)$$

Denote d_{000} to be the number of form (69) and $m_2 = \text{fix}(d_{000}/(d - d_0 - d_{00}))$. Then replace continuously segments (69) to the form:

$$D05 = [1000001110000000], D06 = [0000000111000001] \quad (70)$$

by interval m_2 until all absent zero 5-runs have been generated if possible. If we cannot find enough J05's and J06's defined by (69) to increase the number of zero 5-runs, we need to find continuously segments which have the following form:

$$J07 = [11000000000001]. \quad (71)$$

Denote d_{0000} to be the number of form (71) and $m_3 = \text{fix}(2d_{0000}/(d - d_0 - d_{00} - d_{0000}))$. Then replace continuously segments (71) to the form:

$$D07 = [1000001000001]. \quad (72)$$

by interval m_3 until all absent zero 5-runs have been supplied if possible.

(c) If $d = l_1 - 156 \geq 1$. Find the segments which have the following forms:

$$J11 = [0111110111111], J12 = [1111110111110]. \quad (73)$$

Denote d_1 to be the number of forms (73) and $m = \text{fix}(d_1/d)$. Then replace continuously segments (73) to the forms:

$$D11 = [0111110111111], D12 = [1111110111110] \quad (74)$$

by interval m until all excess one 5-runs have been deleted if possible. If we cannot find enough J11's and J12's defined by (73) to reduce the number of one 5-runs, we need to find continuously the segments which have the following forms:

$$J13 = [11111100111110], J14 = [01111100111111]. \quad (75)$$

Denote d_{11} to be the number of forms (75) and $m_1 = \text{fix}(d_{11}/(d - d_1))$. Then replace continuously segments (75) to the forms:

$$D13 = [11111100111110], D14 = [01111100111111] \quad (76)$$

by interval m_1 until all surplus one 5-runs have been deleted if possible. If we cannot find enough J13's and J14's defined by (75) to reduce the number of one 5-runs, we need to find continuously segments which have the following forms:

$$J15 = [111111000111110], J16 = [011111000111111]. \quad (77)$$

Denote d_{111} to be the number of forms (77) and $m_2 = \text{fix}(d_{111}/(d - d_1 - d_{11}))$. Then replace continuously segments (77) to the forms:

$$D15 = [11111100111110], D16 = [01111100111111] \quad (78)$$

by interval m_2 until all surplus one 5-runs have been deleted if possible. If we cannot find enough J15's and J16's defined by (77) to reduce the number of one 5-runs, we need to find continuously segments which have the following form:

$$J17 = [0111110111110]. \quad (79)$$

Denote d_{1111} to be the number of forms (79) and $m_3 = \text{fix}(2d_{1111}/(d - d_1 - d_{11} - d_{111}))$. Then replace continuously segments (79) to the forms:

$$D17 = [0011111111110] \quad (80)$$

by interval m_3 until all surplus one 5-runs have been deleted if possible.

(d) If $d = 156 - l_1 \geq 1$. Find the segments which have the following forms:

$$J11 = [01111110111111], J12 = [11111101111110]. \quad (81)$$

denote d_1 to be the number of forms (81) and $m = \text{fix}(d_1/d)$. Then replace continuously segments (81) to the forms:

$$D11 = [01111101111111], D12 = [11111110111110] \quad (82)$$

by interval m until all absent one 5-runs have been supplied if possible. If there do not exit enough j01's and j02's defined by (81) to increase the number of one 5-runs, we need to find continuously the segments which have the following forms:

$$J13 = [011111100111111], J14 = [111111001111110]. \quad (83)$$

Denote d_{11} to be the number of forms (83) and $m_1 = \text{fix}(d_{11}/(d - d_1))$. Then replace continuously segments (81) to the forms:

$$D13 = [011111001111111], D14 = [111111100111110] \quad (84)$$

by interval m_1 until all absent one 5-runs have been generated if possible. If there do not exit enough j13's and j14's defined by (83) to increase the number of one 5-runs, we need to find continuously the segments which have the following forms:

$$J15 = [0111111000111111], J16 = [1111110001111110]. \quad (85)$$

Denote d_{111} to be the number of forms (85) and $m_2 = \text{fix}(d_{111}/(d - d_1 - d_{11}))$. then replace continuously segments (85) to the forms:

$$D15 = [0111110001111111], D16 = [1111111000111110] \quad (86)$$

by interval m_2 until all absent one 5-runs have been generated if possible. If there do not exit enough j15's and j16's defined by (85) to increase the number of one 5-runs, we need to find continuously the segments which have the following form:

$$J17 = [00111111111110]. \quad (87)$$

Denote d_{1111} to be the number of forms (87) and $m_3 = \text{fix}(d_{1111}/(d - d_1 - d_{11} - d_{111}))$. Then replace continuously segments (87) to the forms:

$$D17 = [01111101111110] \quad (88)$$

by interval m_3 until all absent one 5-runs have been generated if Possible.

Remark The 6th procedures might "reduce" the uniformities of other runs in order to guarantee better uniformity of 5-runs.

(7) Reducing the numbers of Long Runs.

We find over 26-bit segments which have the following forms:

$$\begin{cases} J0 = [000000000000 \cdots 00000000000000] \\ J1 = [111111111111 \cdots 11111111111111] \end{cases} \quad (89)$$

and then the center zero and one by one and zero

$$D0 = [000000000 \cdots 010 \cdots 0000000000]$$

and

$$D1 = [111111111 \cdots 101 \cdots 111111111],$$

respectively.

(8) Improving the Poker test results for S .

First let us remember the Poker test in the FIPS 140-2 test [7]. Assume the length of a binary sequence S is 20 000. Divide S into 5,000 consecutive 4-bit segments. Denote $f(i)$ to be the number of each 4-bit with decimal value $i - 1$, where $1 \leq i \leq 16$. Then calculate the statistic test item:

$$N = \frac{16}{5000} \sum_{i=1}^{16} f(i)^2 - 5000. \quad (90)$$

If

$$2.16 < N < 46.7.$$

Then S passes the Poker test.

Second if S fails to pass the Poker test, we give an approach to improve the Poker test results for S .

Denote the 16 different 4-bit segments as

$$a_{1,i}a_{2,i}a_{3,i}a_{4,i}, i = 0, 1, \cdots, 15.$$

Denote $f(a_{1,i}a_{2,i}a_{3,i}a_{4,i})$ to be the number of the decimal vale of the 4-bit segment $a_{1,i}a_{2,i}a_{3,i}a_{4,i}$

Case 1. If $N \leq 2.16$.

If

$$f(a_{1,i}a_{2,i}a_{3,i}a_{4,i}) < f(a_{4,j}a_{3,j}a_{2,j}a_{1,j}).$$

Then substitute $a_{1,i}a_{2,i}a_{3,i}a_{4,i}$ by $a_{4,j}a_{3,j}a_{2,j}a_{1,j}$. If

$$f(a_{1,i}a_{2,i}a_{3,i}a_{4,i}) > f(a_{4,j}a_{3,j}a_{2,j}a_{1,j}).$$

Then substitute $a_{4,j}a_{3,j}a_{2,j}a_{1,j}$ by $a_{1,i}a_{2,i}a_{3,i}a_{4,i}$.

In this way substitute, in S , continuously the $a_{1,i}a_{2,i}a_{3,i}a'_{4,i}$ s and the $a_{1,j}a_{2,j}a_{3,j}a'_{4,j}$ s until $N > 2.16$ if possible.

Case 2. If $46.7 \geq N$

If

$$f(a_{1,i}a_{2,i}a_{3,i}a_{4,i}) > f(a_{4,j}a_{3,j}a_{2,j}a_{1,j}).$$

Then substitute $a_{1,i}a_{2,i}a_{3,i}a_{4,i}$ by $a_{4,j}a_{3,j}a_{2,j}a_{1,j}$. If

$$f(a_{1,i}a_{2,i}a_{3,i}a_{4,i}) < f(a_{4,j}a_{3,j}a_{2,j}a_{1,j}).$$

Then substitute $a_{4,j}a_{3,j}a_{2,j}a_{1,j}$ by $a_{1,i}a_{2,i}a_{3,i}a_{4,i}$

In this way substitute, in S , continuously the $a_{1,i}a_{2,i}a_{3,i}a'_{4,i}$ s and the $a_{1,j}a_{2,j}a_{3,j}a'_{4,j}$ s until $N < 46.7$ if possible.

In summary, we complete the procedures for improving the randomness of pseudorandom sequences based on FIPS 140-2 test criterions.

Remark The 6th and 7th procedures will effect negative results to procedures (2)-(5). In practical applications, we may meet the situation that one improving procedure (1-runs to 5-runs) makes the later runs become worse such that them failed to pass FISP 140-2 test. In this case, we will give up the improving procedures for 1-runs to 5-runs, that is, only implement the improving procedures for Monobit test, Pork test and Long Runs test.

2.1 A new chaotic generalized synchronization system

In order to design the CPRNG, we introduce a new 8-dimensional chaotic generalized synchronization system (8DCGSS). Firstly, the driving system of the 8DCGSS is the combination of the Henon map, the logistic map and the tube map [10]:

$$\begin{aligned}\mathbf{X}(k+1) &= \begin{pmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ x_4(k+1) \end{pmatrix} \\ &= \begin{cases} 1 - 1.4x_1(k)^2 + x_2(k) \\ 0.3x_1(k) \\ 3.15x_3(k)(1 - x_3(k)) \\ 6.8x_4(k)(1 - x_4(k))^2 \end{cases}\end{aligned}\quad (91)$$

Written in a compact form, it is

$$\mathbf{X}(k+1) = F(\mathbf{X}(k)). \quad (92)$$

The calculated Lyapunov exponents of this system are $\{0.2543, 0.1411, -0.7363, -1.6339\}$. Therefore the 8DCGSS will be a hyper-chaotic system.

Secondly, construct an invertible matrix

$$A = \begin{pmatrix} 2 & -1 & -5 & -4 \\ 2 & -5 & -6 & 8 \\ -6 & 7 & -5 & -1 \\ -4 & -6 & -1 & -6 \end{pmatrix}, \quad (93)$$

and define a transformation $H : \mathbb{R}^4 \longrightarrow \mathbb{R}^4$ as follows

$$H(\mathbf{X}) = A\mathbf{X} \triangleq (h_1(\mathbf{X}), h_2(\mathbf{X}), h_3(\mathbf{X}), h_4(\mathbf{X}))^T. \quad (94)$$

Let $q(\mathbf{X}, \mathbf{Y}) = 1/8(A\mathbf{X} - \mathbf{Y})$. The driven system has the form

$$\mathbf{Y}(k+1) = A[F(\mathbf{X}(k))] - q(\mathbf{X}(k), \mathbf{Y}(k)). \quad (95)$$

Then $q(\mathbf{X}, \mathbf{Y})$ makes the zero solution of error equation (96) to be asymptotically stable.

$$\mathbf{e}(k+1) = H(\mathbf{X}(k+1)) - \mathbf{Y}(k+1) = q(\mathbf{X}, \mathbf{Y}). \quad (96)$$

By the Theorem given in [8], the driving system (91) and the driven system (95) are in generalized synchronization with respect to the transformation H .

Now, select the following initial conditions:

$$\mathbf{X}(0) = (0, 0, 0.5, 0.2)^T, \mathbf{Y}(0) = A\mathbf{X}(0). \quad (97)$$

The chaotic orbits of the state variables x_1, x_2, x_3, x_4 and y_1, y_2, y_3, y_4 for the first 2,000 iterations are shown in Figs.1(a) - (d) and Figs.1(e) - (h), respectively.

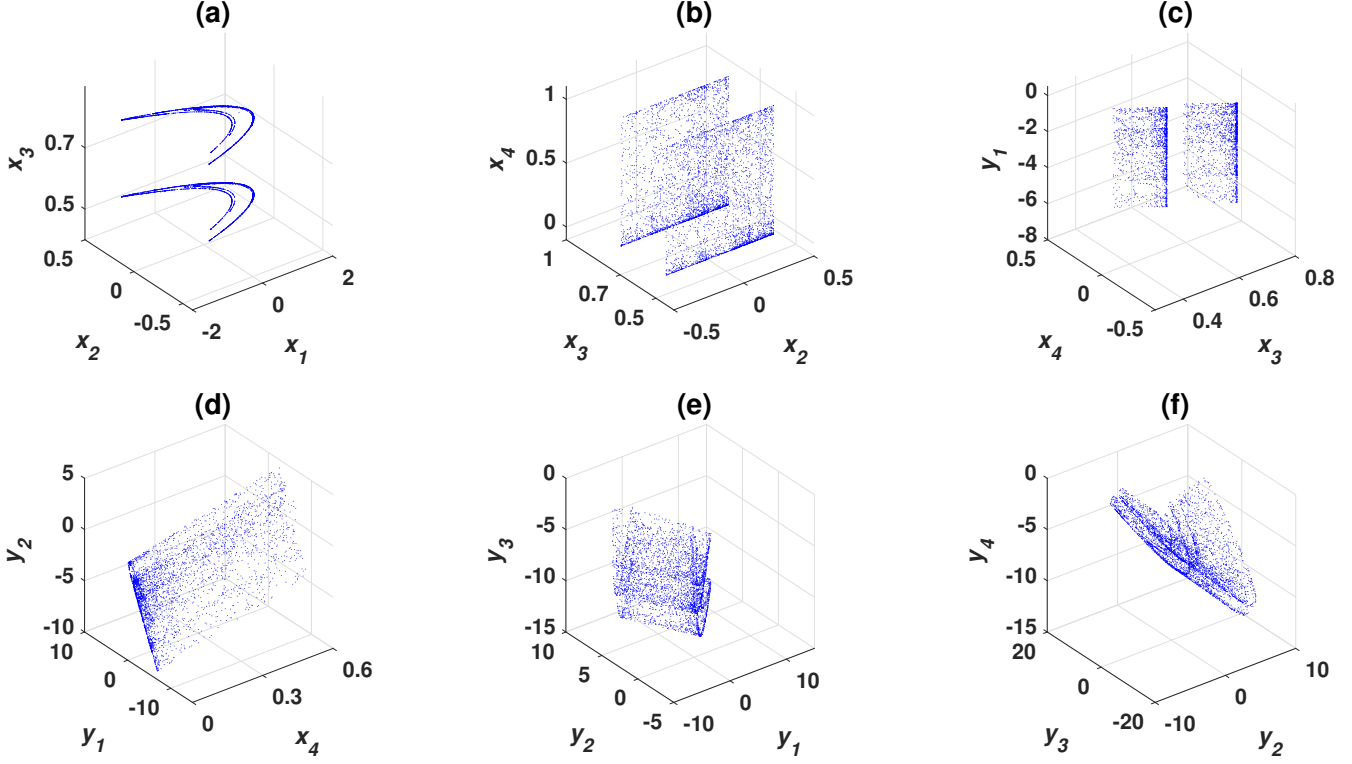


Figure 1: Chaotic trajectories of variables: (a) $x_1(k) - x_2(k) - x_3(k)$, (b) $x_1(k) - x_2(k) - x_4(k)$, (c) $x_3(k) - x_4(k) - y_1(k)$, (d) $x_4(k) - y_1(k) - y_2(k)$, (e) $y_1(k) - y_2(k) - y_3(k)$, and (f) $y_2(k) - y_3(k) - y_4(k)$.

The evolution of state variables: $k - x_1(k)$, $k - x_2(k)$, $k - x_3(k)$ and $k - x_4(k)$ are shown in Figs. 2(a) - (d). The evolution of state variables: $k - y_1(k)$, $k - y_2(k)$, $k - y_3(k)$ and $k - y_4(k)$ are shown in Figs. 3(a) - (d). Figures 4(a)-(d) show that the state variables $\mathbf{X}(k)$ and $\mathbf{Y}(k)$ are in GS with respect to the transformation $H = A$, as the theory predicts. Extensive simulations show that the dynamic behaviors of the GS system have chaotic attractor characteristics.

2.2 A chaotic pseudorandom number generator

Now we transform the chaotic streams generated by the 8DCGSS to binary key streams (denoted by K). Denote

$$\begin{cases} C_1 &= \{x_2(k) | k = 1, 2, \dots, N + 500\} \\ C_2 &= \{y_3(k) | k = 1, 2, \dots, N + 500\}, \end{cases} \quad (98)$$

where x'_1 's and y'_4 's are defined by (91) and (95).

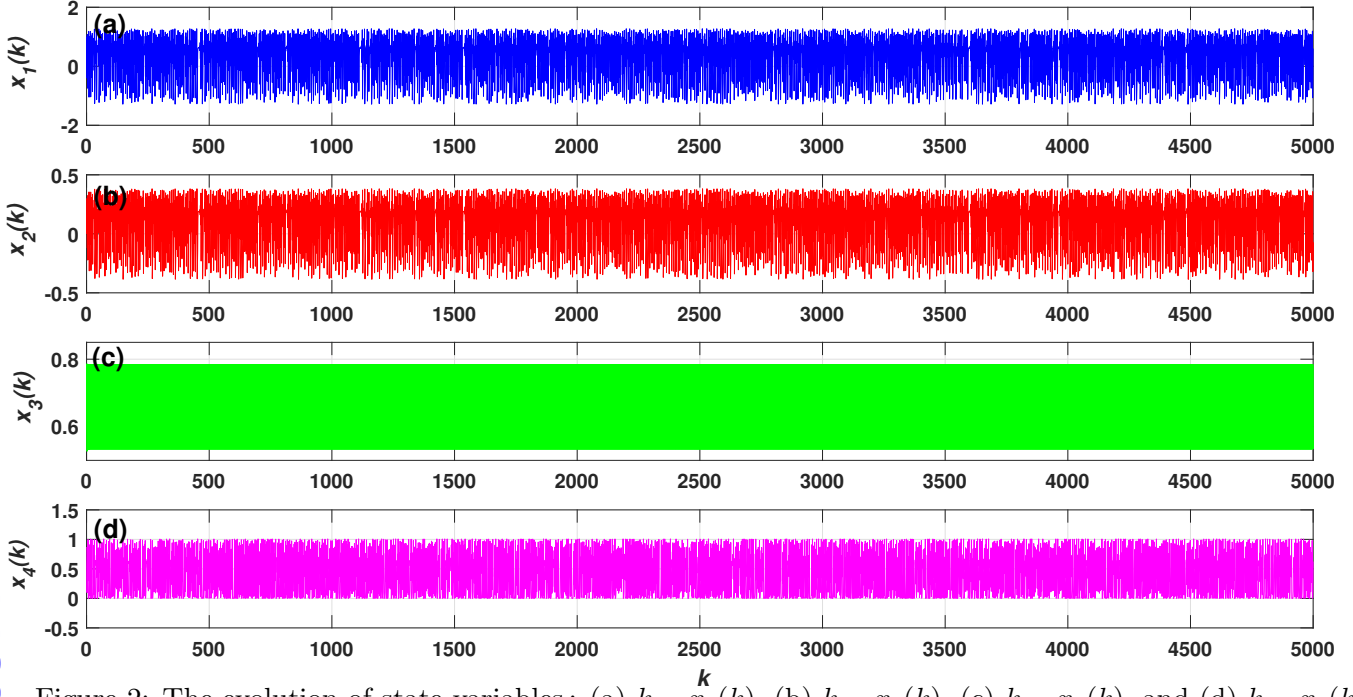


Figure 2: The evolution of state variables: (a) $k - x_1(k)$, (b) $k - x_2(k)$, (c) $k - x_3(k)$, and (d) $k - x_4(k)$.

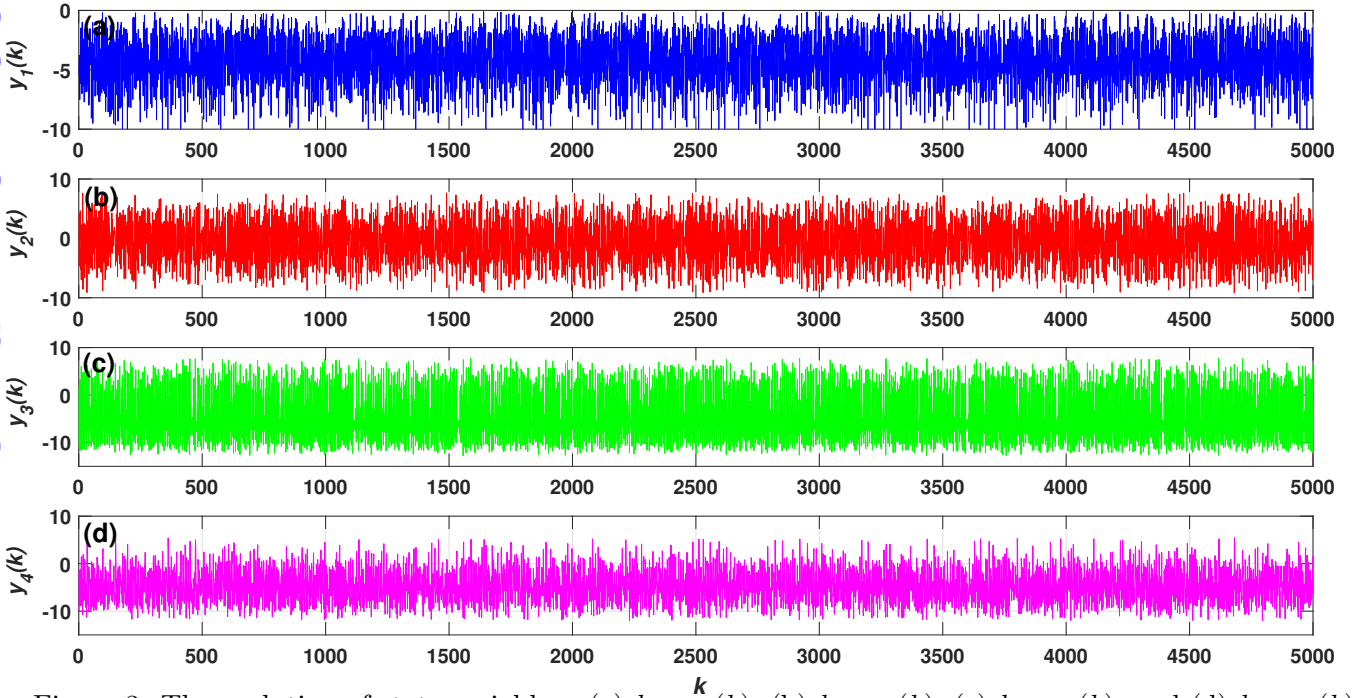


Figure 3: The evolution of state variables: (a) $k - y_1(k)$, (b) $k - y_2(k)$, (c) $k - y_3(k)$, and (d) $k - y_4(k)$.

Let

$$\begin{cases} M_1 = \text{mod} \left(\text{round} \left(\frac{10^{15}(C_1 - \min(C_1))}{\max(C_1) - \min(C_1)} \right), 2^8 \right) \\ M_2 = \text{mod} \left(\text{round} \left(\frac{10^{15}(C_2 - \min(C_2))}{\max(C_2) - \min(C_2)} \right), 2^8 \right), \end{cases} \quad (99)$$

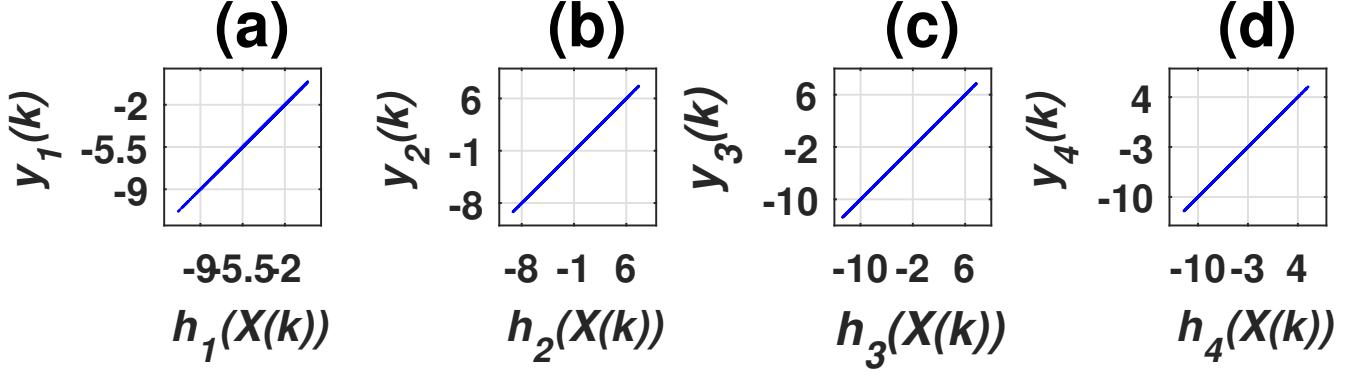


Figure 4: The state vectors \mathbf{X} and \mathbf{Y} are in generalized synchronization with respect to the transformation H . (a) $h_1(\mathbf{X}(k)) - y_1(k)$, (b) $h_2(\mathbf{X}(k)) - y_2(k)$, (c) $h_3(\mathbf{X}(k)) - y_3(k)$, and (d) $h_4(\mathbf{X}(k)) - y_4(k)$.

Now we discard the first 500 iterated sequences and define

$$S_1 = \text{mod}(C_1 + C_2, 2^8), S = \text{dec2bin}(S_1)', \quad (100)$$

where dec2bin is a Matlab command which converts decimal integer to its binary representation. Then the keystream \mathbf{K} with length N is defined by

$$\mathbf{K} = S(:). \quad (101)$$

In summary, we design a CPRNG based on chaotic streams (98) and the transformations (99) - (101)

The seeds of the CPRNG are the initial condition (97) of the GS systems with random perturbations, which can be chosen via random number generators.

The key set parameters of the CPRNG include the initial conditions initial condition $(\mathbf{X}(0), \mathbf{Y}(0))$, and the parameters of matrix (93): $A = (\alpha_{i,j})$. It can be proved that if the perturbation matrix $\Delta = (\delta_{i,j})$ satisfies

$$|\delta_{i,j}| < \frac{1}{4\|A^{-1}\|} \approx 1.3091,$$

then matrix $A + \Delta$ is still invertible. Therefore the CPRNG has $4 + 4 + 16$ key parameters denoted by

$$\mathbf{K}_s = \{k_1, k_2, \dots, k_{24}\}. \quad (102)$$

Let the key set be perturbed by

$$\mathbf{K}_s(\Delta) = \mathbf{K}_s + [\delta_1, \delta_2, \dots, \delta_{24}] \quad (103)$$

where

$$10^{-15} \leq |\delta_i| \leq 10^{-1}, \quad i = 1, \dots, 24.$$

Now we compare the difference between the key stream S with 20 000 bits length generated by the key set (102) and the key streams S'_p s generated by the perturbed key set (103), respectively. The comparison results are shown in the third column in Table 1, where DC denotes the statistic value of the different codes, and CC the one of the correlation coefficients.

The results show that the average percentage of different codes is about 50.029%, which is very closed to

Table 1: The statistic data for the percentages of the codes of the key stream CPRNG variations between S and $S'_p s$ and S and $S'_m s$.

Item	SV	$S'_p s$	$S'_m s$
DC	min	48.835%	48.89
	mean	50.029 %	49.997
	max	51.075 %	51.055%
CC	min	1.03e-4	1.12e-4
	mean	0.5624	0.5657
	max	2.3301	2.3635

the ideal value of 50%. And the mean of the correlation coefficients is about 0.5624. Now, compare the same key stream S with the 1000 key streams $S'_m s$ generated by the Matlab command `randi([0 1], 1, 20000)`. The comparison results are shown in the fourth column in Table 1. Observed that the average percentage of different codes is about 49.997% and the mean of the correlation coefficients is about 0.5657. The results suggest that the key stream S has no significant correlations with the perturbed key streams $S'_p s$.

The Matlab platform uses double-precision decimal computations, which means that each computed decimal number has 16 bits accuracy. In summary, the key space of the CPRNG is larger than $2 \times 10^{14 \times 24} > 2^{1117}$.

2.3 Improvements and measures of pseudorandomness for keystreams generated from 4 PRNGs

This subsection will use the procedures posed in Section 2 to improve the pseudorandomness for the 1000 keystreams with 20000 bits generated by 4 PRNGs, respectively. Then using FIPS 140-2 pseudorandomness test and GFIPS 140-2 pseudorandomness test evaluates the pseudorandomness of the original and the improved 1000 keystreams with length 20 000 bits, respectively.

FIPS 140-2 test issued by INST [7] consists of four sub-tests: Monobit Test, Poker Test, Runs Test and Long Runs Test. Each test needs a single stream of 20 000 one and zero bits from the keystream generator. Any failure in the first three tests means that the corresponding quantity of the sequences falls out the *required intervals* listed in the second column in Table 2. If there are runs of length 26 or more, the Long Runs test fail.

In a previous paper ([9]), we have pointed out that the *required intervals* of the Monobit test, the Poker test and the Runs test correspond to the confident intervals with significant levels: $\alpha = 10^{-4}$, 10^{-4} and $\alpha \approx 1.6 \times 10^{-7}$ (approximately), respectively. The confident intervals for the runs test with significant levels: $\alpha = 1.6 \times 10^{-7}$ are listed in the third column in Table 2. Observe that they are slightly different from the corresponding *required intervals* of FIPS 140-2 runs test. Therefore the required intervals of the run tests given by FIPS 140-2 are not reasonable.

The accepted intervals of the runs test with significant levels: $\alpha = 10^{-4}$ are listed in the 4th column in Table 2. We called the *accepted intervals* as GFIPS 140-2 test (criteria). Observe that the *accepted intervals* are much small than the corresponding *required intervals* of the FIPS 140-2 runs test.

According to Golomb's three postulates on the pseudorandomness that ideal pseudorandom sequences should satisfy [6], the ideal values of the first three tests are listed in the 5th column in Table 2.

(1) Improvements and Tests of Pseudorandomness for Keystreams generated from the CPRNG

First, using the CPRNG with perturbed randomly initial condition $(\mathbf{X}(0), \mathbf{Y}(0))$ (see (97)), and the

Table 2: The required intervals of FIPS 140-2. The accepted intervals with significant levels: $\alpha = 1.6 \times 10^{-7}$, and 10^{-4} for Monobit Test, Poker Test and Runs Test. The Golomb's assumptions require values. Here MT, PT, LT, and RT represent the Monobit Test, the Poker Test and the Long Runs Test, and the Runs test. k represents the length of the run of a tested sequence.

Test Item	FIPS 140-2 Standard Required Intervals	$\alpha = 10^{-4}$ Accepted Intervals	$\alpha = 10^{-4}$ Accepted Intervals	Golomb's Postulates
MT	9 725~10 275	9 725~10 275	9 725~10 275	10000
PT	2.16~46.17	2.16~46.17	2.16~46.17	χ^2 DT
LT	< 26	< 26	< 26	—
RT	FIPS 140-2 Standard Required Intervals	$\alpha = 1.6 \times 10^{-7}$ Accepted Intervals	$\alpha = 10^{-4}$ Accepted Intervals	Golomb's Postulates
1	2315 ~2685	2315 ~2685	2 362~2,638	2500
2	1114~1386	1119~1381	1153~1347	1250
3	527~723	532~718	556~694	625
4	240~384	247~378	264~361	313
5	103~209	110~203	122~191	156
6+	103~209	110~203	122~191	156

parameters of matrix (93) in the range $\epsilon \in [10^{-16}, 10^{-2}]$ generates 1000 keystreams with 20 000 bit length.

Second, using FIPS 140-2 criterions and GFIPS 140-2 criterions tests the 1000 keystreams. The results show that all sequences successfully pass FIPS 140-2 test and there are 17 sequences failing to pass GFIPS 140-2 test (in three sequences, two different test items failed to pass the test, respectively). The calculated results are listed in the 3rd column in Table 3, in which the statistic results of all tests are described by mean values \pm standard deviation (Mean \pm SD).

Third, using the 7 procedures posed in Sections 2 deals with the 1000 keystreams. And then using FIPS 140-2 criterions and GFIPS 140-2 criterions tests the improved 1000 keystreams. The results show that all 1000 sequences pass the FIPS 140-2 test, and there are only 1 sequence failing to pass GFIPS 140-2 test. The calculated results are listed in the 3rd column in Table 4.

The numbers of the runs of the original and the improved 1000 streams which fail to pass GFIPS 140-2 criterions are shown in the 3rd column in Table 5. Observe that the pseudorandomness of the improved 1000 keystreams is significantly better than that of the original 1000 keystreams.

(2) Improvements and Tests of Pseudorandomness for Keystreams Generated from Matlab PRNG

First, using the Matlab command `randi([0,1], 1000, 20000)` generates 1000 keystreams with 20 000 bit length. And then using FIPS 140-2 criterions and GFIPS 140-2 criterions tests the 1000 keystreams. The results show that all sequences successfully pass FIPS 140-2 test and there are 30 sequences failing to pass GFIPS 140-2 test (in one sequence, two different test items failed to pass the test). The calculated results are listed in the 4th column in Table 3,

Second, using the 7 procedures posed in Section 2 deals with the 1000 keystreams. And then using FIPS 140-2 criterions and GFIPS 140-2 criterions tests the improved 1000 keystreams. The results show that the all improved

1000 keystreams pass both FIPS 140-2 test and GFIPS 140-2 test . The calculated results are listed in the 4th column in Table 4.

Table 3: The Mean \pm SD of FIPS 140-2 tested values for 1000 key streams with 20 000 bits generated by the CPRNG, the Matlab PRNG, the RC4 algorithm, and the m-Sequence, respectively. Here, MT, PT and LT represent Monobit Test, Poker Test, and Long Runs Test.

Test item	bits	CPRNG Mean \pm SD	Matlab Mean \pm SD	RC4 Mean \pm SD	m-sequence Mean \pm SD
MT	0	10003 \pm 72.06	9999.8 \pm 71.00	10001 \pm 69.78	9997.7 \pm 68.10
	1	9996.60 \pm 72.06	10000.0 \pm 71.00	9999.1 \pm 69.78	10002.3 \pm 68.10
PT	–	15.16 \pm 5.428	15.12 \pm 5.84	14.90 \pm 5.43	14.07 \pm 12.23
LT	0	13.62 \pm 1.86	13.67 \pm 1.91	13.63 \pm 1.84	13.23 \pm 1.68
	1	13.58 \pm 1.88	13.67 \pm 1.89	13.67 \pm 1.82	13.55 \pm 1.79
k	bits	Run Test	Run Test	Run Test	Run Test
1	0	2498 \pm 48.41	2500.2 \pm 44.54	2497.9 \pm 46.69	2500.4 \pm 44.55
	1	2501.4 \pm 44.73	2500.4 \pm 46.20	2497.7 \pm 46.18	2500.7 \pm 44.45
2	0	1251.8 \pm 31.72	1251.0 \pm 31.55	1250.1 \pm 31.42	1250.3 \pm 32.72
	1	1250.3 \pm 31.98	1249.8 \pm 31.44	1251.6 \pm 30.90	1249.4 \pm 28.16
3	0	625.5 \pm 23.01	624.3 \pm 23.41	625.0 \pm 22.26	625.5 \pm 23.13
	1	624.79 \pm 23.15	625.1 \pm 23.03	623.6 \pm 23.32	625.4 \pm 20.50
4	0	313.1 \pm 16.37	311.8 \pm 16.87	312.2 \pm 17.10	312.2 \pm 17.38
	1	312.0 \pm 16.42	312.3 \pm 15.54	312.2 \pm 17.20	312.4 \pm 16.06
5	0	156.1 \pm 12.19	156.7 \pm 11.83	156.6 \pm 12.01	156.7 \pm 12.06
	1	156.0 \pm 11.79	155.9 \pm 12.13	157.0 \pm 12.09	156.6 \pm 11.34
6+	0	155.9 \pm 11.94	156.1 \pm 11.86	156.5 \pm 11.46	155.5 \pm 10.19
	1	156.0 \pm 11.26	156.6 \pm 12.17	156.2 \pm 11.65	156.2 \pm 10.72

Table 4: The Mean \pm SD of FIPS 140-2 tested values for the improved 1000 key streams with 20 000 bits generated by the CPRNG, the Matlab PRNG, the RC4 algorithm, and the m-Sequence, respectively.

Test item	bits	CPRNG Mean \pm SD	Matlab Mean \pm SD	RC4 Mean \pm SD	m-sequence Mean \pm SD
MT	0	10001 \pm 2.31	10000 \pm 2.40	10000 \pm 2.29	10000.4 \pm 2.13
	1	9999.5 \pm 2.31	9999.5 \pm 2.40	9999.5 \pm 2.29	9999.6 \pm 2.13
PT	–	13.07 \pm 5.32	12.85 \pm 5.39	12.79 \pm 5.04	11.76 \pm 9.14
LT	0	13.65 \pm 1.89	13.64 \pm 1.83	13.63 \pm 1.84	13.2 \pm 1.60
	1	13.54 \pm 1.87	13.63 \pm 1.90	13.61 \pm 1.75	13.5 \pm 1.87
k	bits	Run Test	Run Test	Run Test	Run Test
1	0	2502.2 \pm 9.46	2502.0 \pm 9.19	2501.9 \pm 9.08	2501.1 \pm 14.21
	1	2502.0 \pm 8.25	2501.9 \pm 9.09	2501.8 \pm 10.48	2500.1 \pm 15.16
2	0	1250.4 \pm 8.05	1250.3 \pm 7.66	1250.2 \pm 7.75	1248.8 \pm 10.01
	1	1251.5 \pm 8.30	1251.4 \pm 8.02	1251.8 \pm 8.64	1251.2 \pm 7.20
3	0	623.6 \pm 13.78	624.1 \pm 13.41	624.4 \pm 13.57	623.3 \pm 14.10
	1	625.3 \pm 15.10	625.2 \pm 13.60	624.8 \pm 14.68	623.9 \pm 13.95
4	0	314.9 \pm 16.53	314.8 \pm 16.35	315.2 \pm 16.61	315.2 \pm 13.95
	1	313.1 \pm 16.43	312.9 \pm 16.35	313.2 \pm 16.27	313.4 \pm 15.42
5	0	156.2 \pm 11.56	156.0 \pm 11.68	155.2 \pm 11.46	156.1 \pm 12.20
	1	153.3 \pm 11.55	153.1 \pm 11.31	154.0 \pm 11.43	154.2 \pm 10.44
6+	0	155.0 \pm 10.81	155.0 \pm 11.17	155.3 \pm 10.88	155.8 \pm 9.80
	1	157.1 \pm 10.69	157.5 \pm 10.66	156.8 \pm 10.78	157.4 \pm 9.47

The numbers of the runs of the original and the improved 1000 streams which fail to pass GFIPS 140-2 criterions are shown in the 4rd column in Table 5. Observe that the pseudorandomness of the improved 1000 keystreams is significantly better than that of the original 1000 keystreams.

(3) Improvements and Tests of Pseudorandomness for Keystreams from RC4 Algorithm

Table 5: The numbers of the runs of the original and the improved 1000 streams which fail to pass GFIPS 140-2 criterions. Here FN represents failed number.

Test	bits	CPRNG FN	Matlab FN	RC4 FN	m-sequence FN
		original/improved	original/improved	original/improved	original/improved
MT	0	0/0	0/0	0/0	7/0
	1	0/0	0/0	0/0	7/0
PT	–	0/0	0/0	0/0	25/6
LT	0	0/0	0/0	0/0	0/0
	1	0/0	0/0	0/0	0/0
k	bits	Run Test	Run Test	Run Test	Run Test
1	0	2/0	2/0	3/0	7/2
	1	0/0	6/0	1/0	3/1
2	0	2/0	2/0	1/1	1/0
	1	2/0	2/0	3/0	0/0
3	0	1/0	3/0	4/1	0/0
	1	1/1	0/0	2/1	2/0
4	0	0/0	6/0	4/0	0/0
	1	1/0	0/0	8/1	8/0
5	0	1/0	3/0	2/1	3/0
	1	4/0	3/0	1/1	0/0
6+	0	5/0	0/0	1/0	6/0
	1	1/0	4/0	1/0	0/0

The Matlab program for RC4 algorithm is described in Fig. 5. First, using the above command generates 1000 keystreams with 20 000 bit length. And then using FIPS 140-2 criterions and GFIPS 140-2 criterions tests the 1000 keystreams. The results show that all 1000 sequences pass FIPS 140-2 test, and there are 30 sequences failing to pass GFIPS 140-2 test (in one sequence, two different test items failed to pass the test). The calculated statistic results are listed in the 5th column in Table 3.

Second, using the 7 procedures presented in Section 2 deals with the 1000 keystreams. And then using FIPS 140-2 criterions and GFIPS 140-2 criterions tests the improved 1000 keystreams. The results show that the all improved 1000 keystreams pass FIPS 140-2 test. There are 6 sequences failing to pass GFIPS 140-2 test. The calculated results are listed in the 5th column in Table 4.

The numbers of the runs of the original and the improved 1000 streams which fail to pass GFIPS 140-2 criterions are shown in the 5th column in Table 5. Observe that the pseudorandomness of the improved 1000 keystreams is significantly better than that of the original 1000 keystreams.

```

clear all
L=8;
RC=randi([0,2^L-1], 1000,2^L);
Keystream(1000,20000)=0;
L=8;
for J=1:1:1000
    j=0;
    S=[0:2^L-1];
    for i=1:2^L
        j=mod(j+S(i)+RC(J,i),2^L);
        Sk=S(j+1);
        S(j+1)=S(i);
        S(i)=Sk;
    end
    j=0;i=0;
    C(1,20000/8)=0;
    for l=1:20000/8
        i=mod(i+1,2^L);
        j=mod(j+S(i+1),2^L);
        Sk=S(j+1);
        S(j+1)=S(i+1);
        S(i+1)=Sk;
        C(l)=S(mod(S(j+1)+S(i+1),2^L)+1);
    end
    M=(dec2bin(C))';
    M=(M(:)-48)';
    Keystream(J,:)=M;
end

```

Fig. 5. RC algorithm written by Matlab program.

(4) Improvements and Tests of Pseudorandomness for Keystreams Generated from an m -Sequence

The m -sequences are binary bit sequences generated via maximal linear feedback shift registers. The occurrences of 0 and 1 in an m -sequence with one period length satisfy Golomb's three postulates on the randomness.

Every m -sequence corresponds to a polynomial representation. Now we choose an m -sequence with the form

$$x^{20} + x^3 + 1. \quad (104)$$

The 1000 keystreams with 20 000 bit generated by this m -sequence can be implemented by the Matlab program shown in Fig. 6.

```

clear all
X1=randi(1000,20);
Keystream(1000,20000)=0;
for J=1:1000
    k=1;
    while k<=20000 %
        uu=mod( (X1(20)+X1(3)),2);
        X(k)=uu;
        X1=[uu X1(1:end-1)];
        k=k+1;
    end
    Keystream(J,:)=X;
end

```

Fig. 6. *m*-sequence written by Matlab program.

First, using FIPS 140-2 criterions and GFIPS 140-2 criterions tests the 1000 *m*-sequence keystreams with 20 000 bits. The results show that there are 28 sequences failing to pass FIPS 140-2 test (in two sequences, two different test items failed to pass the tests, respectively; in five sequences, three different test items failed to pass the tests, respectively); there are 44 sequences failing to pass GFIPS 140-2 test (in six sequences, two different test items failed to pass the tests, respectively; in two sequences, three different test items failed to pass the tests, respectively; in three sequences, six different test items failed to pass the tests, respectively) . The calculated statistic results are listed in the 6th column in Table 3. Such unexpected results imply that the pseudorandomness of the first 20 000 bits of *m*-sequences is much worse than that of other three PRNGs.

Second, using the 7 procedures posed in Section 2 deals with the 1000 keystreams. And then using FIPS 140-2 criterions and GFIPS 140-2 criterions tests the improved 1000 keystreams. The results show that that there are 14 sequences failing to pass FIPS 140-2 test, and there are 17 sequences failing to pass GFIPS 140-2 test (in one sequence, two different test items failed to pass the tests; in three sequences, three different test items failed to pass the tests, respectively). The calculated results are listed in the 6th column in Table 4. The numbers of the runs of the original and the improved 1000 streams which fail to pass GFIPS 140-2 criterions are shown in the 6th column in Table 5. Observe that the pseudorandomness of the improved 1000 keystreams is significantly better than that of the original 1000 keystreams.

In summary, the improved keystreams of the four pseudorandom number generators can significantly increase the pseudorandomness of the corresponding original keystreams. In particularly, the numbers of failing to pass the two test can be decreased significantly (see Tables 5 and 6).

Table 6: Failing sequence numbers. Here, FIPS/GFIPS represent the failing numbers of the sequences which cannot pass FIPS 140-2 test and GFIPS 140-2 test.

	Original Sequences	Improved Sequences
PRNG	FIPS/GFIPS	FIPS/GFIPS
CPRNG	0/17	0/1
Matlab	0/30	0/0
RC4	0/30	0/6
m-Sequence	28/45	14/8

2.4 Improvements and measures of pseudorandomness for keystreams with length $1e5$ generated from 4 PRNGs

This subsection will use the procedures posed in Section 2 to improve the pseudorandomness for the 1000 keystreams with 100 000 bits generated by the 4 PRNGs, respectively. Then use FIPS 140-2 randomness criterions and GFIPS 140-2 pseudorandomness criterions evaluate the pseudorandomness of the original and the improved 1000 keystreams, respectively.

In a previous paper ([9]), we have pointed out that the *required intervals* of the Monobit test, the Poker test and the Runs test correspond to the accepted intervals with significant levels: $\alpha = 10^{-4}$, 10^{-4} and $\alpha \approx 1.6 \times 10^{-7}$ (approximately), respectively. The accepted intervals for the runs test with significant levels: $\alpha = 1.6 \times 10^{-7}$ are listed in the third column in Table 2. Observe that they are slightly different from the corresponding *required intervals* of FIPS 140-2 runs test. Therefore the required intervals of the run tests given by FIPS 140-2 are not reasonable.

As the statements given in 3.3 subsection and the formulas posed in paper ([9]), the accepted intervals of FIPS 140-2 test and GFIPS 140-2 test for 100 000 bite codes are listed in Table 7.

Now, using the 7 procedures posed in Section 2 deals with the 1000 keystreams with length 100 000 bits.

(1) For the CPRNG, there were 2 original sequences and 37 original sequences failing to pass FIPS 140-2 test and GFIPS 140-2 test, respectively. There were 0 improved sequences and 2 improved sequences failing to pass FIPS 140-2 test and GFIPS 140-2 test, respectively.

(2) For the Matlab PRNG, there were 2 original sequences original sequences failing to pass FIPS 140-2 test and and 31 original sequences failing to pass GFIPS 140-2 test (in five sequences, two different items failed to pass the tests, respectively), respectively. There were 0 improved sequences and 3 improved sequences failing to pass FIPS 140-2 test and GFIPS 140-2 test, respectively.

(3) For the RC4 algorithm, there were 0 original sequences failing to pass FIPS 140-2 test, and 34 original sequences and GFIPS 140-2 test (in one sequences, two different items failed to pass the test, respectively). There were 0 improved sequences and 4 improved sequences failing to pass FIPS 140-2 test and GFIPS 140-2 test, respectively.

(4) For the m -sequence, there were 15 original sequences and 37 original sequences failing to FIPS 140-2 test and GFIPS 140-2 test, respectively. There were 13 improved sequences and 18 improved sequences failing to pass FIPS 140-2 test and GFIPS 140-2 test, respectively.

Table 7: The accepted intervals for 100 000 bit sequences with significant levels: $\alpha = 1.6 \times 10^{-7}$, and 10^{-4} for Monobit Test, Poker Test and Runs Test. The Golomb’s randomness assumptions required values. Here MT, PT, and LT represent the Monobit Test, the Poker Test and the Long Runs Test. k represents the length of the run of a tested sequence.

Test	FIPS 140-2 $\alpha = 10^{-4}$	GFIPS 140-2 $\alpha = 10^{-4}$	Golomb’s Postulates
	Accepted intervals	Accepted intervals	
MT	49 171~50 829	49,171~50 829	50 000
PT	2.16~46.17	2.16~46.17	χ^2 DT
LT	< 26	< 26	—
Runs k	$\alpha = 1.6 \times 10^{-7}$	$\alpha = 10^{-4}$	Golomb’s Postulates
	Accepted intervals	Accepted intervals	
1	12 086 ~12 914	12 192~12 808	12 500
2	5957~6543	6033~6467	6250
3	2918~3332	2971~3279	3125
4	1416 ~ 1709	1454 ~1671	1563
5	678~ 885	704 ~ 858	781
6+	678~ 885	704 ~ 858	781

Table 8: Failing numbers. Here, FIPS/GFIPS represent the failing numbers of the sequences which cannot pass the FIPS 140-2 test and the GFIPS 140-2 test.

	Original SQ	Improved SQ
PRNG	FIPS/GFIPS	FIPS/GFIPS
CPRNG	2/37	0/2
Matlab	2/31	0/3
RC4	0/34	0/4
m-Sequence	15/37	0/5

The calculated results are summarized in Table 8. The detailed statistic results are summarized in Table 9 - Table 11. Observe that the first three PRNGs can generate sound pseudorandom key streams in the meanings in the sense of Golomb’s randomness postulates. The presented new approach can increase significantly the pseudorandomness of the keystreams generated the four PRNGs. The m-sequence cannot generate good key streams whose lengths are less than 100 000 bits.

2.5 Improvements and measures of pseudorandomness for keystreams with length 10^6 generated from 4 PRNGs

This subsection will use the procedures posed in Section 2 to improve the pseudorandomness for the 1000 keystreams with $1e6$ bits generated by the 4 PRNGs, respectively. Then use FIPS 140-2 pseudorandomness criterions and GFIPS 140-2 pseudorandomness criterions to evaluate the pseudorandomness of the original and the improved 1000 keystreams, respectively.

Table 9: The Mean \pm SD of FIPS 140-2 tested values for 1000 key streams with length $1e5$ bits generated by the CPRNG, the Matlab PRNG, the RC4 algorithm, and the m-Sequence, respectively. Here, MT, PT and LT represent Monobit Test, Poker Test, and Long Runs Test.

Test item	bits	CPRNG Mean \pm SD	Matlab Mean \pm SD	RC4 Mean \pm SD	m-sequence Mean \pm SD	Golomb's Postulates
MT	0	49 999 \pm 155.29	500 07 \pm 158.82	50 000 \pm 155.99	50 007 \pm 146.37	50 000
	1	50 001 \pm 155.29	49 993 \pm 158.82	50 000 \pm 155.99	49 993 \pm 146.37	
PT	–	15.09 \pm 5.35	15.37 \pm 5.66	14.91 \pm 5.29	14.81 \pm 9.75	$\chi^2 DT$
LT	0	15.86 \pm 1.79	15.90 \pm 1.86	15.97 \pm 1.90	15.86 \pm 1.73	–
	1	16.01 \pm 1.93	15.95 \pm 1.85	15.85 \pm 1.87	16.01 \pm 1.68	
k	bits	Run Test	Run Test	Run Test	Run Test	–
1	0	12 502 \pm 106.04	12 495 \pm 106.54	12 496 \pm 101.96	12 502 \pm 102.5	12 500
	1	12 503 \pm 107.72	12 499 \pm 103.44	12 496 \pm 105.41	12 504 \pm 103.1	
2	0	6247.7 \pm 70.65	6252.1 \pm 72.07	6249.4 \pm 68.52	6250.3 \pm 75.54	6250
	1	6246.3 \pm 70.83	6251.9 \pm 70.37	6250.3 \pm 75.54	6249.7 \pm 66.76	
3	0	3124.4 \pm 51.33	3125.6 \pm 51.80	3124.8 \pm 51.91	3123.1 \pm 53.28	3125
	1	3125.1 \pm 50.06	3124.6 \pm 50.11	3121.4 \pm 52.77	3125.9 \pm 46.60	
4	0	1563.2 \pm 36.40	1563.6 \pm 38.00	1561.6 \pm 37.03	1563.7 \pm 50.16	1563
	1	1562.7 \pm 37.87	1562.3 \pm 36.92	1564.5 \pm 37.66	1561.2 \pm 32.146	
5	0	780.73 \pm 27.24	781.8 \pm 27.08	782.43 \pm 26.42	781.58 \pm 23.97	781
	1	781.16 \pm 27.72	778.64 \pm 26.17	782.64 \pm 27.00	780.13 \pm 34.48	
6 ⁺	0	781.54 \pm 26.34	780.55 \pm 27.39	781.28 \pm 25.40	781.44 \pm 22.04	781
	1	781.75 \pm 26.60	782.12 \pm 27.59	781.09 \pm 26.05	780.95 \pm 23.79	

Table 10: The Mean \pm SD of FIPS 140-2 tested values for improved 1000 key streams with length $1e5$ generated by the CPRNG, the Matlab PRNG, the RC4 algorithm, and the m-Sequence, respectively.

Test item	bits	CPRNG Mean \pm SD	Matlab Mean \pm SD	RC4 Mean \pm SD	m-sequence Mean \pm SD	Golomb's Postulates
MT	0	50 000 \pm 5.54	50 000 \pm 5.38	49 999 \pm 4.10	49 999 \pm 4.83	50 000
	1	50000 \pm 5.54	50000 \pm 5.38	4289.2 \pm 495.82	50001 \pm 4.83	
PT	–	13.07 \pm 5.16	13.08 \pm 5.26	14.55 \pm 5.43	12.92 \pm 8.07	$\chi^2 DT$
LT	0	15.81 \pm 1.79	15.86 \pm 1.82	15.97 \pm 1.92	15.83 \pm 1.71	–
	1	15.97 \pm 1.86	15.92 \pm 1.84	15.79 \pm 1.84	16.22 \pm 1.70	
k	bits	Run Test	Run Test	Run Test	Run Test	–
1	0	12 505 \pm 28.97	12 504 \pm 29.67	12 502 \pm 29.918	12 504 \pm 24.77	12 500
	1	12 505 \pm 29.59	12 504 \pm 27.09	12 503 \pm 31.82	12 502 \pm 18.96	
2	0	6246.1 \pm 24.62	6246.1 \pm 26.72	6245.7 \pm 28.31	6241.9 \pm 41.31	6250
	1	6248.2 \pm 28.33	6249.1 \pm 24.55	6247.3 \pm 29.37	6246.3 \pm 26.37	
3	0	3123.8 \pm 23.91	3122.4 \pm 22.85	3123.3 \pm 28.48	3118.4 \pm 27.05	3125
	1	3126.8 \pm 26.06	3125.3 \pm 24.46	3124.6 \pm 27.42	3125.5 \pm 23.16	
4	0	1560.8 \pm 38.23	1560.8 \pm 37.94	1559.0 \pm 40.34	1562.3 \pm 35.90	1563
	1	1557.8 \pm 38.56	1556.2 \pm 37.30	1555.7 \pm 37.908	1553.3 \pm 42.05	
5	0	785.54 \pm 19.91	785.62 \pm 19.32	785.78 \pm 20.33	785.55 \pm 21.37	781
	1	776.03 \pm 21.14	775.60 \pm 20.89	778.06 \pm 20.65	775.64 \pm 26.58	
6 ⁺	0	780.46 \pm 27.08	780.80 \pm 26.62	781.85 \pm 28.60	783.14 \pm 27.06	781
	1	787.65 \pm 25.88	789.61 \pm 25.65	789.10 \pm 84.63	792.72 \pm 27.09	

Table 11: The numbers of the runs of the original and the improved 1 000 streams which fail to pass GFIPS 140-2 criterions. Here FN represents failed number.

Test	bits	CPRNG FN original/improved	Matlab FN original/improved	RC4 FN original/improved	m-sequence FN original/improved
MT	0	0/0	0/0	0/0	0/0
	1	0/0	0/0	0/0	0/0
PT	–	1/0	1/0	0/0	15/0
LT	0	0/0	1/0	0/0	0/0
	1	1/0	0/0	0/0	0/0
k	bits	Run Test	Run Test	Run Test	Run Test
1	0	4/0	3/0	2/0	3/0
	1	8/0	2/0	5/1	0/0
2	0	3/0	0/1	2/1	2/2
	1	1/0	3/0	5/0	0/0
3	0	2/1	3/1	3/1	0/3
	1	2/0	0/0	3/1	0/0
4	0	1/0	5/0	1/0	0/0
	1	2/0	2/0	5/0	0/0
5	0	2/0	3/0	1/0	0/0
	1	5/0	4/1	4/0	0/0
6+	0	3/0	5/0	1/0	0/0
	1	3/1	3/0	2/0	0/0

As the statements given in 3.3 subsection and the formulas prosed in paper ([9]), the accepted intervals of FIPS 140-2 test and GFIPS 140-2 test for 1e6 bite codes are listed in Table 12.

Now, using the 7 procedures posed in Section 2 deals with the 1000 keystreams with length 1e6 bits for the 4 CPRNGs, respectively. (1) For the CPRNG, there were 9 original sequences and 41 original sequences failing to pass FIPS 140-2 test and GFIPS 140-2 test, respectively. There were 0 improved sequences failing to pass FIPS 140-2 test, and 8 improved sequences failing to pass GFIPS 140-2 test (in one sequence, two different items failed to pass the test, respectively), respectively.

(2) For the Matlab PRNG, there were 8 original sequences and 49 original sequences (in three sequences, there are two different items not to pass the tests, respectively) failing to pass FIPS 140-2 test and GFIPS 140-2 test, respectively. There were 0 improved sequences and 8 improved sequences (in two sequences, there are two different items not to pass the tests, respectively) failing to pass FIPS 140-2 test and GFIPS 140-2 test, respectively.

(3) For the RC4 algorithm, there were 12 original sequences and 34 original sequences (in one sequence, there are two different items not to pass the tests) failing to pass FIPS 140-2 test and GFIPS 140-2 test, respectively. There were 1 improved sequences and 3 improved sequences failing to pass FIPS 140-2 test and GFIPS 140-2 test, respectively.

(4) For the m-sequence, there are 9 original sequences failing to pass FIPS 140-2 test, and 41 original sequences failing to pass GFIPS 140-2 test. All improved sequences pass FIPS 140-2 test and GFIPS 140-2 test, respectively.

Table 13 summaries the above results. The detail statistic test results for the 4 PRNGS are summarized in Table 14~16.

From Tables 3, 9 and 14, it follows that the pseudorandomness of the original codes of the CPRNG, the Matlan PRNG, and the RC4 algorithm become better when the lengths of the codes becomes longer (see (mean value + std value - idea value)/(idea value)); the pseudorandomness of original codes of the *m*-sequence become much better when the lengths of the codes equal to 1e6 (see (mean value + std value - idea value)/(idea value)).

From Tables 4, 10 and 15, it follows that the pseudorandomness of the improved codes of the CPRNG, the Matlan PRNG, and the RC4 algorithm become better when the lengths of the codes becomes longer. The

Table 12: The accepted intervals for 100 000 bit sequences with significant levels: $\alpha = 1.6 \times 10^{-7}$, and 10^{-4} for Monobit Test, Poker Test and Runs Test. The Golomb's randomness assumptions required values. Here MT, PT, and LT represent the Monobit Test, the Poker Test and the Long Runs Test. k represents the length of the run of a tested sequence.

Test	FIPS 140-2 $\alpha = 10^{-4}$	GFIPS 140-2 $\alpha = 10^{-4}$	Golomb's Randomness
Item	Accepted intervals	Accepted intervals	Postulates
MT	497 380 ~502 620	498 055 ~501 945	500 000
PT	2.16~46.17	2.16~46.17	χ^2 DT
LT	< 26	< 26	–
	$\alpha = 1.6 \times 10^{-7}$	$\alpha = 10^{-4}$	–
k	Runs Test	Runs Test	–
1	123 690~126 310	124 027~125 973	125 000
2	61 574~63 426	61 812 ~ 63 188	62 500
3	30 595~31 905	30 764 ~31 736	31 250
4	15 162~16 088	15 281~15 969	15 625
5	7485~8140	7569~8056	7813
6+	7485~8140	7569~8056	7813

Table 13: Failing numbers. Here, FIPS/GFIPS represent the failing numbers of the sequences which cannot pass the FIPS 140-2 test and the GFIPS 140-2 test.

PRNG	Original SQ FIPS/GFIPS	Improved SQ FIPS/GFIPS
CPRNG	9/41	0/4
Matlab	8/49	0/8
RC4	12/34	1/3
m-Sequence	9/41	0/0

pseudorandomness of original codes of the m -sequence become excellent when the lengths of the codes equal to $1e6$. Tables 13 - 16 show clearly that the improved codes increase significantly the pass rates of GFIPS 140-2 criterions.

3 Conclusions

The main contributions of this paper are concluded as follows.

- (1) Based on Golomb's assumptions on idea pseudorandom sequences, presented a general approach for improving pseudorandomness of pseudorandom sequences.
- (2) Used the Henon map, the logistic map the tube map and the GS theorem to construct a 8-dimensional generalized synchronization system and implemented numerical simulations.
- (3) Used the 8-dimensional generalized synchronization system designs a chaotic pseudorandom number generator (CPRNG). The key space of the CPRNG is larger than 2^{1117} .

Table 14: The Mean \pm SD of FIPS 140-2 tested values for 1000 key streams with length 1e6 bits generated by the CPRNG, the Matlab PRNG, the RC4 algorithm, and the m-Sequence, respectively. Here, MT, PT and LT represent Monobit Test, Poker Test, and Long Runs Test.

Test item	bits	CPRNG Mean \pm SD	Matlab Mean \pm SD	RC4 Mean \pm SD	m-sequence Mean \pm SD	Golomb's Postulates
MT	0	500 013 \pm 511.31	500 003 \pm 506.44	500 002 \pm 484.84	500 000 \pm 112.26	500 000
	1	499 987 \pm 511.31	499 997 \pm 506.44	499 998 \pm 484.84	500 000 \pm 112.26	
PT	–	15.155 \pm 5.411	15.11 \pm 5.69	14.54 \pm 5.16	11.56 \pm 2.96	$\chi^2 DT$
LT	0	19.23 \pm 1.84	19.27 \pm 1.85	19.28 \pm 1.88	18.95 \pm 0.23	–
	1	19.29 \pm 1.96	19.19 \pm 1.83	19.27 \pm 1.95	19.90 \pm 0.44	
k	bits	Run Test	Run Test	Run Test	Run Test	Run Test
1	0	124 996 \pm 318.03	124 997 \pm 333.97	125 000 \pm 330.79	125 000 \pm 74.95	125 000
	1	125 002 \pm 330.64	124 993 \pm 314.96	125 014 \pm 337.81	124 999 \pm 71.17	
2	0	62 501 \pm 228.89	62 508 \pm 230.51	62 505 \pm 288.61	62 499 \pm 53.41	62 500
	1	62 503 \pm 231.00	62 518 \pm 221.76	62 494 \pm 225.59	62 501 \pm 45.70	
3	0	31 249 \pm 163.4	31 253 \pm 166.43	31 253 \pm 161.39	31 250 \pm 37.59	31 625
	1	31 247 \pm 165.6	31 246 \pm 164.71	31 247 \pm 162.37	31 250 \pm 33.16	
4	0	15 632 \pm 118.26	15 623 \pm 118.15	15 620 \pm 118.75	15 624 \pm 30.04	15 625
	1	15 625 \pm 121.04	15 627 \pm 117.47	15 622 \pm 118.38	15 624 \pm 24.98	
5	0	7810.6 \pm 85.31	7814.2 \pm 82.26	7814.4 \pm 84.62	7812.6 \pm 17.59	7183
	1	7811.1 \pm 85.88	7811.7 \pm 87.13	7809.4 \pm 84.61	7812.9 \pm 21.82	
6+	0	7811.9 \pm 84.35	7808.9 \pm 82.72	7811.9 \pm 84.45	7813.0 \pm 16.51	7813
	1	7812.0 \pm 84.30	7809.0 \pm 84.92	7817.5 \pm 82.30	7812.9 \pm 16.05	

Table 15: The Mean \pm SD of FIPS 140-2 tested values for improved 1000 key streams with length 1e6 generated by the CPRNG, the Matlab PRNG, the RC4 algorithm, and the m-Sequence, respectively. Here, MT, PT and LT represent Monobit Test, Poker Test, and Long Runs Test.

Test item	bits	CPRNG Mean \pm SD	Matlab Mean \pm SD	RC4 Mean \pm SD	m-sequence Mean \pm SD	Golomb's Postulates
MT	0	499 998 \pm 10.86	499 998 \pm 12.48	499 999 \pm 11.38	5000 000 \pm 0	500 000
	1	500 001 \pm 10.86	500 001 \pm 12.48	500 001 \pm 11.38	5000 000 \pm 0	
PT	–	12.82 \pm 5.12	12.77 \pm 5.21	12.46 \pm 4.90	11.48 \pm 2.93	$\chi^2 DT$
LT	0	19.18 \pm 1.75	19.22 \pm 1.74	19.19 \pm 1.70	18.95 \pm 0.25	–
	1	19.18 \pm 1.78	19.10 \pm 1.70	19.19 \pm 1.72	19.90 \pm 0.43	
k	bits	Run Test	Run Test	Run Test	Run Test	–
1	0	125 001 \pm 101.12	125 000 \pm 97.81	125 004 \pm 84.75	125 002 \pm 3.63	12 5000
	1	125 000 \pm 100.69	125 003 \pm 91.02	125 001 \pm 105.62	125 000 \pm 1.02	
2	0	62 499 \pm 98.39	62 501 \pm 103.84	62 497 \pm 98.56	62 500 \pm 4.69	62 500
	1	62 506 \pm 111.45	62 507 \pm 95.14	62 506 \pm 95.76	62 501 \pm 4.99	
3	0	31 236 \pm 85.61	31 236 \pm 91.26	31 235 \pm 83.67	31 248 \pm 5.85	31 250
	1	31 265 \pm 80.83	31 264 \pm 85.79	31 270 \pm 79.85	31 253 \pm 7.91	
4	0	15 650 \pm 116.14	15 653 \pm 121.87	15 645 \pm 120.20	15 628 \pm 49.35	15 625
	1	15 599 \pm 117.99	15 596 \pm 116.47	15 597 \pm 120.93	15 621 \pm 43.19	
5	0	7819.6 \pm 44.63	7822.1 \pm 42.84	7822.7 \pm 43.89	7813.7 \pm 4.34	7813
	1	7800.9 \pm 54.76	7799.5 \pm 54.23	7796.0 \pm 56.54	7810.6 \pm 8.50	
6+	0	7797.2 \pm 86.14	7794.3 \pm 87.42	7800.1 \pm 86.37	7813.6 \pm 4.33	7813
	1	7832.9 \pm 78.92	7835.9 \pm 81.82	7834.8 \pm 82.33	7810.6 \pm 8.49	

Table 16: The numbers of the runs of the original and the improved 1000 streams with length $1e6$ which fail to pass GFIPS 140-2 criterions. Here FN represents failed

Test	bits	CPRNG FN original/improved	Matlab FN original/improved	RC4 FN original/improved	m-sequence FN original/improved
MT	0	0/0	0/0	0/0	0/0
	1	0/0	0/0	0/0	0/0
PT	—	0/0	1/0	0/0	0/0
LT	0	3/0	3/0	6/0	3/0
	1	6/0	3/0	5/0	6/0
k	bits	Run Test	Run Test	Run Test	Run Test
[0pt]1	0	0/1	6/0	1/1	0/0
	1	5/0	1/0	2/0	5/0
2	0	3/1	3/3	1/0	3/0
	1	1/1	3/0	1/0	1/0
3	0	2/2	3/1	0/0	2/0
	1	2/1	5/3	1/0	2/0
4	0	2/0	5/0	1/0	2/0
	1	5/0	3/1	2/1	5/0
5	0	6/2	1/0	5/0	6/0
	1	5/1	5/1	3/1	5/0
6+	0	1/0	5/0	5/0	1/0
	1	0/1	4/1	2/0	0/0

- (4) The numerical simulations show that the correlations between the keystream S generated by the CPRNG and the 1000 keystream S'_p s generated by the CPRNG with perturbed keys are similar to those between S and 1000 keystreams generated by the Matlab PRNG. This implies that the keyspace of the CPRNG is large enough to against brute-force attacks.
- (5) Our approach can increase significantly the pseudorandomness of the keystreams generated the four PRNGs. In particularly, the numbers of the improved pseudorandom sequences failing pass the GFIPS-140 test can be decreased significantly (see Tables 3-6, Tables 8 -11 and Tables 13 -16) .
- (6) The m-sequence with period 2^{20} cannot generate good key streams whose lengths are less than 100 000 bits.

In summary, the proposed approach provides a new tool for improving the pseudorandomness of pseudorandom sequences. Using the presented approach can obtain high quality pseudorandom sequences. Researches along this line are promising.

Note

This version has be submitted to the proceedings of the 14th Chaotic Modeling and Simulation International Conference, Heraklion, Crete, Greece, 13 – 16 June, 2023.

References

- [1] K. Binder and D. W. Heermann, *Monte Carlo Simulation in Statistical Physics, An Introduction (4th edition)*, 2002. Springer.
- [2] M. Senator, Synchronization of two coupled escapement-driven pendulum clocks, *J. Sound and Vibration*, 291(3): 566–603, 2006.

- [3] S. Wegenkittl. Gambling Tests for Pseudorandom Number Generator. *Mathematics and Computers in Simulation*, 55, 281–288, 2001.
- [4] D. E. Knuth, *The Art of Computer Programming, Seminumerical Algorithms (3rd ed) Vol. 2*, Boston, Addison-Wesley, 1997.
- [5] N. Ferguson, B. Schneier, T. Kohno, *Cryptography Engineering: Design Principles and Practical Applications*, Wiley Publishing, 2010..
- [6] S. W. Golomb. *Shift Register Sequences*. Revised edition, CA: Aegean Park, Laguna Hills, 1982.
- [7] INST. *FIPS PUB 140-2, Security Requirements for Cryptographic Modules*. Gaithersburg, MD, INST, 2001.
- [8] H. Zang, L. Min, G. Zhao, A Generalized Synchronization Theorem for Discrete-time Chaos System with Application in Data Encryption Scheme, *in Proc of the 2007 Int. Conf. On Communications, Circuit and Systems*, Kokura, Fukora, Japan, Vol. II, July, 325–329, 2007.
- [9] L. Min, T. Chen, H. Zang, Analysis of FIPS 140-2 Test and Chaos-Based Pseudorandom Number Generator, *Chaotic Modeling and Simulation*, 2, 273–280, 2013.
- [10] X. Yang, L. Min, X. Wang, A cubic map chaos criterion theorem with applications in generalized synchronization based pseudorandom number generator and image encryption, *Int. J. Bifurcat. Chaos*, 25(5), 053104: 1–9, 2015.